An End-User Programming Paradigm for Pervasive Computing Applications

Jeannette S Chin, Vic Callaghan, Graham Clarke, Intelligent Inhabited Environments Group, *UK*

?

***Abstract*— In this paper we present a variant of end-user programming called PiP (Pervasive interactive Programming) which offers non-technical end-users the possibility to configure and customize sets of coordinating pervasive devises without the need to employ conventional programming methods. In this approach end-users "show" the system their required behaviour via natural physical interaction with the environment. The paper also describes the architectural components and presents a user evaluation.**

Keywords— **Pervasive Computing, End-user Programming, Programming-by-Example, Disaggregation, Digital Homes.**

I. INTRODUCTION

The end-user programming paradigm is characterised by techniques that allow users of application programs to create "programs" without any technical expertise [4]. There are many ways to achieve this, for example through use of existing programming abstractions in which code is replaced by graphical representations to form customised "scripting or macro languages" that a user can use to create a desired functionality. Smith, in the mid-seventies, introduced an approach called Programming-by-Example, where functionality was demonstrated directly via concrete examples by the end-users, rather than derived indirectly from the use of programming abstractions [7]. This was developed further by Lieberman in the nineties [10]. Traditionally, end-user programming was aimed solely at creating applications that ran on single desktop computing environments.

This paper presents a variant of the end-user programming paradigm, primarily targeting pervasive computing environments. It employs a "*show-me-by-example*" approach allowing non-technical end-users to "program" their environment to suit their particular needs. The end-users are neither required to write program code, nor follow a rigid sequential list of actions in order to achieve the results. All the end-user needs to do, is simply to *show* the system the required functional behaviour by demonstrating the required physical actions within the environment. We called this method, Pervasive Interactive Programming (PiP); UK Patent No: GB 0523246.7. This paper builds on earlier reported work [1] [2] [3], presenting a prototype and evaluation.

II. MOTIVATIONS

The motivation behind PiP was to create a system that engendered a sense of trust and empowered user creativity by maximizing user control and operational transparency whilst enabling users to "program" their own environment, without the need for detailed technical knowledge.

To date the majority of the research directed at this area has focused on streamlining the use of the input languages or metaphor-based GUI interfaces, aimed at simplifying the use of the applications for the users. Currently most end-user programming tools for pervasive applications are based on the procedural programming metaphor and require the user to mentally manipulate constructs that would be familiar to most programmers (albeit in a graphical or macro form) thereby placing a significant cognitive load on the user. We have been inspired by the ease with which people perform daily routine tasks (eg. switching on the lights when a room gets dark, muting the TV sound when a telephone rings etc) and so we decided to direct our approach at finding a way of programming that was natural and mimicked familiar practices as much as possible, without the need for the users to follow a set of rigid logical sequences of actions.

III. PERVASIVE INTERACTIVE PROGRAMING

Pervasive Interactive Programming (PiP) is aimed primarily at *end users* in a *service-rich* pervasive environment. We assume that services are offered from networked devices supported by underlying protocol layers which are not described in this paper. PiP provides a platform that utilises the physical user space as the programming environment thereby providing a natural and familiar mechanism for the user to "program" the functionality that they require to suit their particular needs. Thus, with a minimum effort, a non-technical end-user is able to customise the functionality of coordinating groups of pervasive computing devices that could usually only be achieved by conventional programming.

A. Background Concept

1) Pervasive Device and Applications

A pervasive environment is heavily populated with network aware devices and services. It is centred on the concept of services that provide functions to accomplish particular tasks. The success of these tasks is partly attributed to the ability of

a device to communicate their internal states (the term "device" refers to any network application that is able to either initiate or react to commands relating to a service). With a supporting software framework, these services are discoverable, and accessible to the environment in which they reside. An example of a supporting framework is Universal Plug and Play (UPnP)[1].

2) The Deconstructed Model – Virtual Device

As devices and their services in pervasive environments are discoverable and accessible, a number of possibilities emerge. For instance by aggregating sets of services it becomes possible to form "virtual devices" that offer higher level composite services. We refer to such communities or "virtual devices" as **M**eta**App**liance**s (MAps)** and the approach as the deconstructed device model.

3) MetaApplicances (MAp)

The concept of a MAp is a core concept in PiP. From a logical perspective, a MAp has primitive properties and a collection of *Rules* that determine the behaviour of the coordinating devices and, as a consequence, the environment, which is the end user's personal space. Rules are essentially a marriage of 2 different types of actions, namely 'Antecedent' (condition) and 'Consequent' (action). Each action (whether it is an 'Antecedent' or a 'Consequent') has the property of a "virtual device". The 'Antecedent' of a Rule can be described as "if" while the 'Consequent' of a Rule can be described as "then". A Rule can contain 0-n 'Antecedents' and 1-n 'Consequents', and a MAp legally can contain 0-n Rules (as Rules can be added later by the end user).

MAps are a non-terminating *process* and require no specific user expertise for their formation. They are created *under the directions of end users* to provide the sort of behaviour they like. They can be represented graphically and be visible to the user who created them, either at the time of creation or later when they can be retrieved, shared, executed, or removed on demand. Until a MAp is terminated, it will retain the functionalities that the user originally created (ie. it is a continually running process).

4) PiP System Architecture

PiP is designed to work in real time within a pervasive environment. The communication between PiP, the end user and the environment is via an eventing mechanism, thus PiP has an event-based object-oriented asynchronous architecture. Unlike macro languages, where sequence of actions is significant, PiP assumes the logical sequence of actions is not important. It employs a rule policy to maintain a MAp process in which "a set of conditions are satisfied if the conditions defined within the context of this set are all satisfied". PiP leverages UPnP™ technology as its middleware and communication protocol, enabling simple and robust connectivity among devices and PCs. It has modular framework, comprising six core modules, which work together to support real-time network computation. The core modules are:

a. "Interaction Execution Engine" (IEE) – this module has a network control point and is responsible for device discovery, service events subscription, and performing network action requests.

b. "Eventing Handler" (EH) – this module acts like a "middle-man", responsible for interpreting low-level network events (eg device discovery), device service events (eg service state changes) and high-level events that generate from "PiPView" caused by the user interactions. Its main role is to communicate events between interested modules.

c. "Knowledge Engine" (KE) - this module is responsible for assembling and instantiating a "virtual device" (ie a MAp) before storing them in the Knowledge Bank. It is also responsible for updating the device's current status, as well as maintaining an up-to-date version of the Knowledge Bank.

d. "Real-time MAp Maintenance Engine" (RTMM) — is a process that maintains the records of current and previously created MAps.

e. "Real-Time Rule Formation Engine" (RTRF) – this module is responsible for assembling rules based on user interactions within the "demonstration" mode[2].

f. "GUI" – A graphical interface called "PiPView" that the user can use to make inspections of the environment, compose/delete Maps/Rules etc, interact with the system and control physical environment.

B. How Does the System Work?

This section illustrates how the system actually works when put together. In PiP, apart from the "PiPView" GUI, other networked devices can be regarded as user interface, since users interact with them during the demonstration process. Examples are: networked dimmer lights, networked telephone, network entertaining systems, network fridge etc (Figure 1). Using these devices, users can interact intuitively and naturally with the environment and the metaphor for programming their environment is thus very simple. The user creates a MAp ie. the "program" that captures the functionalities of the environment, by showing the system the functionalities that the MAp should have via simple familiar interaction e.g by using a wall switch to turn on a light etc., and PiP will do the rest for the user.

A MAp is created by the user "dragging & dropping" device representations through PiPView. A MAp can be given collective functionality by the user demonstrating the required behaviour by engaging in physical activities using the *real* devices, previously selected via PiPView. In PiP, the user can choose when to inform the system they are ready to begin to program (show) the device functionalities by using any of the three methods: (1) physically interacting with the devices

[1] UPnP network technology allows devices to offer their services to network clients. More details UPnP forum at: http://www.upnp.org/

[2] A "demonstration" mode begins when the user clicks "ShowMe" button and end when user clicks "Done!" button.

themselves, (2) using a UI control panels (3) a combination of the above two the choice being left to the user.

Figure 1. PiP meets pervasive environment

Based on the actions of the user, the pervasive devices generate appropriate events and pass them to the network and PiP encodes this information as a set of rules with two parts; an antecedent (conditions) and consequent (resulting action) as it "listens" and "captures" the user's action as shown .

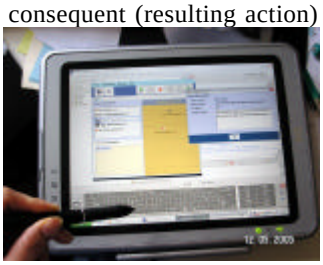

Figure 2. PiP on tablet view

In PiP the user is given as much freedom as possible, allowing antecedents and consequents to be formed in any number and order ie. the user is not required to follow a rigid order. To execute a MAp, the user needs only to drag the MAp graphical representation and drop it into a "play" button located at the top of the PiPView. To terminate a MAp the user simply clicks on the "stop" button.

IV. RELATED WORK

Much attention has been paid to research in this area, the most relevant to PiP being Media Cubes [6] which offers a tangible interface for programming an environment where each face of a cube is represented by a set of program structures. "Programming operations" are achieved by turning the appropriate face of the cube towards the target device. Humble [9] uses a jigsaw, metaphor, enabling users to "snap" together puzzle-like graphical representations as a way of building applications. Truong's CAMP project [8] places the end-users at the centre of the design experience by using a fridge magnet metaphor, together with a pseudo-natural language interface that collectively enables end-users to realize context-aware pervasive applications in their homes. The Alfred project [5] utilises a macro programming approach to enable a user to compose a program via "teaching-by-example" using verbal or physical interactions. Whilst these are very imaginative and useful approaches, for our particular vision, they are either not flexible enough to support the end users' intuitive physical interactions or place a high cognitive load upon the users (eg. utilising methods such as macros requires users to adhere to a strict ordering of instructions or otherwise the system will fail).

V. END USER EVALUATION

An end user evaluation was carried out in the iDorm2 at the University of Essex[3], a two-bedroom apartment built to be an experimental pervasive computing environment. Five sets of pervasive devices were created for the evaluations- (1) a bed light, (2) a desk-light, (3) telephone, (4) a sofa and (5) an MP3 player. All devices were run on UPnP middleware network.

A. Evaluation Design and Procedure

Our objectives for the evaluation were, broadly, to see how easy end users found PiP for programming their environments. In connection with this a questionnaire was designed to explore the users attitude towards "conceptual", "user control", "cognitive loading", "information presentation", "affective experience" and "future potential of PiP". In addition, a user interview was conducted. . The evaluation sought to provide open ended tasks giving the end users as much creative freedom as possible as this was one key enabling property PiP provides.

Eighteen participants (10 females and 8 males) drawn from a diverse set of backgrounds (eg housewives, students, secretaries, teachers etc) participated in the evaluation. All participants had some minimal computing experience ie. they knew how to use a mouse. Whilst 21.3% of the participants had a very good knowledge of programming, 57.4% of them had none at all. During the evaluations, PiP was set-up to run on a winXP tablet PC that connected to the iDorm2 network via a 802.11g WIFI access point. Each trial was preceded by a 20-minutes training session to allow participant to familiarise themselves with the system. The task for the evaluation was that the participant should use PiP to program the pervasive environment to behave in the way they wanted. No specific type of behaviour for the environment was set for the evaluation, rather the participants were free to create one (or more) of their own. Clearly, with only a 5 devices available the possibilities were somewhat limited but, for example, one user designed a "teletainment" MAp that reacted to a ringing telephone in a way that was dependent on whether an MP3 file was being played, and where the user was sitting. No time limit was set and assistance was provided where needed.

Figure 3. A user demonstrates her actions via physically interacts with the devices

Following completion of the tasks, a questionnaire with a scale of responses ranging from "Strongly Agree" through to

[3] http://iieg.essex.ac.uk/idorm2/index.htm

"Strongly Disagree" was administrated to measure the participants' subjective judgements of PiP. Participants rated a total of seventeen statements covering the six usability dimensions: described above. Data was analysed using SPSS[4].

VI. RESULTS

Results showed that 83% of participants were able to use PiP to program their environment with little or no assistance, although the time taken to accomplish these tasks varied from participant to participants. Of the three methods available for demonstrating examples 11% of the participants chose to program their environment via wholly GUI controls while 72% of them used physical interactions with the environment, the rest (17%) used a mixture of both approaches. Although PiP does not require logical sequence when programming MAps, 33% of the participants expressed the view that they found it easier to think using logical sequence and decided to conduct their trials that way. The remainder of the participants (77%) focused on the task by creating the functionalities within the environment rather than attending to the logical sequence. The study also revealed that none of the participants found it difficult to understand the basic principles of the system.

VII. CONCLUSION AND FUTURE WORK

This paper has described our research into an end-user programming paradigm for pervasive computing applications. We have successfully implemented a "proof of concept" system called PiP, using an event-based modular architecture which enables non-technical end-users to program the environment functionality they require within a pervasive computing environment. Whilst acknowledging that the participants are only a small sample of the population, the initial results are encouraging as they show that PiP served different users well, allowing them to program the environment to suit their needs. Thus, we contend that whilst the user evaluation is relatively small, it has suggested that this approach is usable by non-technical end-users to create their own functionalities in the technology-rich pervasive environments, such as digital homes.

For our future work we hope to conduct further work on knowledge representation at the MAp level. For MAps to be portable across environments it is essential that there is a generic way of describing the capabilities of collectives of devices and services such as based on dComp [5] ontology.

The concept of MAps raises a number of interesting questions. For example domestic appliances can be viewed as a special case of a MAp in which a group of coordinating services are hard wired together by the manufacturer. The notion of end-users being able to "wire together" and program the functionality of their own virtual devices challenges the nature of future appliances; will appliances of the future continue to be pre-packaged physical bundles of services or will a more elemental form of network device emerge?

[4] SPSS at http://www.spss.com/

[5] dComp at http://iieg.essex.ac.uk/dc omp/ont/dev/2004/09/

ACKNOWLEDGMENT

This paper describes PhD research conducted by Jeannette Chin supported by a University of Essex scholarship. It builds on earlier work supported by the DTI Next Wave Technologies and Markets programme as part of the Pervasive Home Environment Networking project. In this we are pleased to express our gratitude to our colleagues, especially Martin Colley, Hani Hagras and Malcolm Lear for their unstinting support.

REFERENCES

[1] Chin J, Callaghan V, Clarke G *"Pervasive Information Systems: Issues for the Individual and Society"*, in Book Pervasive Information Systems published by MB Sharp, August 05

[2] J, Chin et al, Virtual Appliances for Pervasive Computing: A Deconstructionist, Ontology based, programming-By-Example Approach, The IEE IE05, Colchester, UK, 28-29 June 2005.

[3] Chin J, Callaghan V, End-User Programming in Pervasive Computing Environments, The International PSC-05, Monte Carlo Resort, Las Vegas, USA, June 27-30, 2005

[4] Cypher *A, et al,* "Watch What I Do: Programming by Demonstration" The MIT Press, Cambridge, Massachusetts, London, England 1993.

[5] Gajos K., Fox H., Shrobe H., "*End User Empowerment in Human Centred Pervasive Computing*", Pervasive 2002, Zurich, 2002.

[6] Hague, R., et al: Towards Pervasive End-user Programming. In: Adjunct Proceedings of UbiComp 2003 (2003) 169-170

[7] Smith, D. C., "*Pygmalion: A Computer Program to Model & Stimulate Creative Thought*", Stuttgart, Birkhauser Verlag. 1977.

[8] Truong, KN.et al " CAMP: A Magnetic Poetry Interface for End-User Programming of Capture Applications for the Home", Proceedings of Ubicomp 2004, pp 143-160.

[9] Humble, J. et al "Playing with the Bits", User-Configuration of Ubiquitous Domestic Environments, Proceedings of UbiComp 2003, Springer-Verlag, Berlin Heidelberg New York, pp 256-263

[10] Lieberman H. *Your wish is my command*, Morgan Kaufmann Press, 2001

Jeannette S Chin. Jeannette is a member of the IEEE. She has a Civil Architecture background and obtained a first class honours degree in Internet Computing from the University of Essex in UK. Jeannette has a strong interest in pervasive computing and the notion of any-where, any-time, any-person computing. Her research interests include intelligent environments for pervasive computing, semantic information processing, internet related technologies, privacy issues, and user interfaces which encapsulate psychological aspects of human and computer interaction.

Vic Callagham Vic holds a Ph.D in Computing and B.Eng in Electronic Engineering from the University of Sheffield. He is Professor of Computer Science at Essex University. He has contributed to around 100 paper journals, conferences and books and currently, he leads the Inhabited Intelligent Environments Group (IIEG) and is director of the Digital Lifestyles Centre.

Graham Clarke Graham is an honorary fellow at Essex University. His first degree was in (building) Architecture, his MSc was in the Applications of Computing and his PhD was in Psychoanalytic Studies which represent a combination that reflect his commitment to the crucial importance of users in ubiquitous computing environments. He has been involved in intelligent building research in the Department of Computer Science for the past ten years.